
OWNER'S MANUAL

Model 7710

Asynchronous Serial Interface

CALIFORNIA COMPUTER SYSTEMS
APPLE II™ ASYNCHRONOUS SERIAL INTERFACE
MODEL 7710
OWNER'S MANUAL

89000-07710
REV C
Copyright 1980

California Computer Systems
250 Caribbean Drive
Sunnyvale, CA 94086
USA

PREFACE

The CCS Model 7710 Asynchronous Serial Interface gives you the means to interface to your APPLE computer such peripheral devices as video terminals, line printers, paper tape units, and even other computers. As shipped, the 7710 includes everything you need to begin interfacing with asynchronous serial devices except the cable which must be custom-designed to link your peripheral to the 7710.

This manual is intended to provide as complete an understanding as possible of the hardware and software features of the 7710. At the same time, we recognize that many APPLE owners want to be able to plug a board in and use it without having to wade through extensive discussions of hardware and software theory. For those of you in the latter category, Chapter 2 provides all of the information necessary for the set-up and installation of the 7710, while Sections 3.4 through 3.6 tell you how to use the interface once it is installed. Chapter 1, "Theory of Operation," is provided for users curious about the serial interfaces in general and the hardware of the 7710 in specific. Those of you who plan to write your own software will want to read Chapter 3 in its entirety.

A number of addresses referred to in the text depend on the number of the slot in which the 7710 is installed. We use "n" throughout to represent the slot number.

A "-" before a signal name or mnemonic (for example, -DEV SEL) indicates a low-active signal, as does a bar over the name or mnemonic.

CHAPTER 1

THEORY OF OPERATION

1.1 INTRODUCTION

A computer is a very expensive do-nothing unless you can give it data, instruct it what to do with the data, and then have it present the results. To help do this, peripheral devices were designed. But because computers could communicate with peripherals in a number of formats, another kind of device was necessary: interfaces. These devices translate between a computer, which inputs and outputs in one format, and one or more peripherals, which input and output in another format.

There are two major types of computer data transfer: parallel and serial. Parallel interfaces transfer words of data simultaneously on parallel data lines. This type of interface has its limitations: all data bits must be not only transmitted but received simultaneously. This limits the length of the cable connecting the computer and the peripheral; the longer the cable, the more distorted the signals are.

The solution to this problem is serial data transfer. Bits are transmitted one at a time on one wire. The receiving end reassembles the bits into words. To make this scheme work, the sender must have a method to convert parallel data into serial, while the receiver must be able to convert serial data to parallel. This presents a problem for the receiver because the serial data is nothing more than a continuous stream of ones and zeros; how can it tell which group of bits is a word? Two schemes have been devised to solve this problem: synchronous and asynchronous modes of transmission. During synchronous transmission, a pre-defined pattern of synchronization bits is sent out first. When the receiver finds this pattern, it divides the subsequent data into word-length groups. This requires highly precise synchronization of transmitter and receiver. Asynchronous interfaces handle the problem differently. They add a start bit to the front of each word, plus one or more stop bits to the end. The total group of bits is then transmitted one bit at a time. The receiver can identify the data bits, since they are a fixed number of mixed bits appearing between the logical zeros of the start bits and the logical ones of the stop bits.

The receiver must also know how fast the data is being sent. Generally, as long as the receiver expects data at the rate the sender is sending it, the actual rate of transfer does not matter. Common usage has defined many standard signal rates. Usually, they are an even multiple of 75 baud (bits per second--including overhead bits, such as start and stop bits). A few

other rates, 50, 110, and 134.5, are used by the industry giants. RS-232-C does not specify any standard signal rates, but suggests a practical upper limit of 20 kilobaud and indirectly establishes a theoretical upper limit of 50 kilobaud.

It is not enough for computers and peripherals to be able to exchange data; each must also be able to tell when the other is ready to transmit or receive. This is done with "handshaking" signals. Because a wide variety of handshaking schemes are possible, the EIA created the RS-232-C interface specifications to let manufacturers know what to expect. Two "sides" were defined. Because one side of the interface is usually connected to some type of computer terminal, equipment at that end is called Data Terminal Equipment (DTE). Equipment at the other end is called Data Communications Equipment (DCE), because to transmit serial data over long distances via telephone wires, a Modulator/Demodulator, or Modem, is needed. For short distances (less than several hundred feet), modems and telephone wires are not needed, but one side of the interface must be made to think it's DCE.

RS-232-C defines the necessary protocol between the DCE and the DTE for many possible configurations (see section 4.6). Of these, the first five (A, B, C, D, and E) are most commonly used. Configuration A defines a one-way, transmit-only interface, as might be used by a simple serial keyboard. Configuration B is also a one-way, transmit-only interface, but it has more handshaking. A paper tape reader might use this type of

interface. Configuration C is also one-way, but receives only, and might be used by a serial printer.

With Configuration D, we start getting into two-way traffic. Configuration D is HALF DUPLEX; although it can carry traffic both ways, it can only carry one way at a time. When a modem is used in this configuration, it is often called a "two wire modem" because the telephone line is connected with only two wires. This type of interface is not often used. Configuration E is a FULL DUPLEX link. This means that the link can support traffic in both directions at the same time. CRT terminals most often use this type of link. If a modem is used in this configuration, it is called--guess what--a "four wire modem." The 7710 supports all five of these configurations. In addition, a Data Terminal Ready (DTR) handshake line has been provided. Normally used only on synchronous interfaces, this line is also used by some asynchronous printers.

1.2 7710 HARDWARE DESIGN

The 7710 hardware card can be divided into four sections: a) the transmitter/receiver section; b) the baud rate generator; c) the control section; and d) the program memory.

1.2.1 Transmitter/Receiver Section

The major component of this section and the heart of the 7710 is a 6850 Asynchronous Communications Interface Adapter, or ACIA. It performs the parallel-to-serial and serial-to-parallel data conversion, adds start and stop bits when transmitting and removes them when receiving, makes available status information, and controls handshaking with the peripheral or modem. A programmable control register allows specification of word length, number of stop bits, parity type (or inhibition), and clock division ratios, besides initiating transmitter and receiver cycles and enabling or disabling interrupts. This section also includes the necessary line drivers and receivers to comply with the RS-232-C electrical specifications.

On the serial side, the ACIA provides three handshake lines for peripherals/modems. One of these lines, -DATA CARRIER DETECT (-DCD), allows the peripheral to control the ACIA receiver section and initiate an interrupt when it is not ready, in effect allowing the peripheral to tell the computer to slow down. The -DCD input is tied to the inverted Data Terminal Ready (DTR) signal (pin 20) of the RS-232-C connector. The ACIA will stop transmitting when DTR goes low. When DTR goes high, the ACIA will resume transmitting.

The other two handshake signals are -Request to Send (-RTS) and -Clear to Send (-CTS). These signals were named for a DTE device. Since the 7710 is designed to be a DCE device, the roles of these signals are

reversed. -RTS is connected to RS-232-C line CTS, a high on which indicates to the peripheral that the computer has data to transmit. -CTS is connected to RS-232-C line RTS, allowing the peripheral to enable transmission by toggling RTS high.

A 75154 and a 75150 perform the line receiver and line driver functions respectively, translating the standard Transistor-Transistor Logic (TTL) signals of the ACIA to the +5 volt to -5 volt (nominal) signal levels required by RS-232-C. They also provide for fail-safe operation should your interface cable short or disconnect accidentally.

1.2.2 Baud Rate Generator

The ACIA needs a clock to tell it how often to send or sample signals. Different serial devices require data to be clocked at different rates. The 7710 employs a 4702 baud rate generator to supply standard baud rates from 50 to 19200. This chip contains an oscillator for the on-board quartz crystal and a controllable frequency divider circuit. The oscillator generates a highly stable 2.4576 MHz square wave signal. This signal is counted down by a divisor determined by the four switches on the card. The output of the 4702, connected to the ACIA at the Transmit Clock and Receive Clock pins, is 16 times the actual selected baud rate. Thus the ACIA must be programmed to expect a x16 clock.

The 4702, by itself, generates 13 different baud rates, from 50 to 9600,

although the switches allow 16 codes. Two of the codes select 2400 baud. One of the remaining codes is used to select a 19,200 (x16) clock. Normally, the 4702 prescales the 2.4576 Mhz signal by dividing it first by 16 to yield a 9600 (x16) signal. The prescaler, however, allows connection to the signal after it has been divided by 8; thus a 19,200 baud signal is available. The last code selects a signal from pin 24 of the RS-232-C connector, allowing the terminal (or another source) to generate the baud rate. Clock rates of up to 50,000 baud may be used.

1.2.3 Control Section

The control section of the interface consists primarily of a buffer between the ACIA and the computer data lines, and the interrupt arbitration logic. The ACIA has only limited ability to drive the data lines of the computer. To ensure successful data transfer, we have placed a bi-directional line buffer (the 8304B) between the computer and the ACIA. The 8304B consumes a lot of power, though. Since the driver is used only for short periods, a power-down feature has been added. When -DEVICE SELECT goes low, a transistor turns on power to the 8304B, allowing data transfer. When transfer is completed, the computer toggles -DEVICE SELECT high and the transistor turns off power to the 8304B. The 8304B monitors R/-W to determine the direction of data transfer.

The interrupt arbitration logic is one

link in the interrupt daisy chain, which prioritizes peripheral-generated interrupts to ensure that only one device interrupts at a time. If no higher-priority interrupt is in progress (pin 28 high), a low from ACIA output -IRQ will force the -IRQ line to the computer (pin 30) low. Concurrently, pin 23 is pulled low, telling the lower-priority devices that an interrupt is pending, forcing them to wait. After being serviced, the ACIA removes its interrupt request, and the arbitration logic sets the daisy chain signal high again.

Note that the highest priority device must be installed in the leftmost slot in the group 1 through 7. Slot 0 does not support the daisy chain. Empty slots between cards break the daisy chain.

1.2.4 Program Memory

Your computer dedicates 256 bytes of memory space to each peripheral connector. This is sufficient for most interface-unique software. The 7710 includes two 256x4-bit ROMs, enabled by -I/O SEL, which contain a general interface driver routine. As the ROMs consume quite a bit of power, they are equipped with a power-down feature like the 8304B. Should you desire to develop your own software, you may substitute RAMs for the ROMs. If you do so, the memory power-down feature must be disabled and the R/-W line enabled to the RAMs. One jumper wire will do this (see section 2.2). Using RAMs, you can fully develop and test your program before committing it to ROMs.

CHAPTER 2

INSTALLATION AND CHECKOUT

2.1 BOARD AND PERIPHERAL CONFIGURATION

Before you can install and operate your 7710 Asynchronous Interface you will need to configure it and your peripheral for the following.

2.1.1 Baud Rate

Your peripheral manual should tell you at which baud rate or rates the peripheral will operate. If it can handle several baud rates, choose the highest one available which is common to the interface card. Set your terminal for that baud rate, following its instructions. Then set the CCS card switches to match. If your peripheral's baud rate is available as an output and your cable makes the signal available to the 7710 at RS-232-C connector pin 24, you can set the 7710 to the peripheral's baud rate by setting the baud rate switches to the code for External.

With the card in front of you, position it so that the switches are in the upper right-hand corner. Find your baud rate in Table 1, and set the switches as shown for that baud rate (o means to push that side of the rocker switch down).

Table 1. Baud Rate Selector Switch	
	1234
50 Baud ON OFF	o oo o
110 Baud	oooo
150 Baud	o ooo
300 baud	o o oo
1200 Baud	o oo o
2400 Baud	oo oo
4800 Baud	oo o o
19200 Baud	ooo o
75 Baud	oo oo
134.5 Baud	oo o o
200 Baud	o o o o
600 Baud	o o oo
1800 Baud	o o o o
2400 Baud	o ooo
9600 Baud	ooo o
External Clock	oooo

2.1.2 Half/Full Duplex

If your terminal allows you to select between half and full duplex operation, select FULL DUPLEX. Your computer's firmware expects a full duplex keyboard/display. This means that the computer, after receiving a character from the keyboard, sends that character back out to the display, allowing a quick check that the computer received what you wanted it to. When a terminal is in the half duplex mode, it displays the character as it is typed in, then, when the computer echoes, displays it a second time. For example, if you typed in RUN with the terminal in a half duplex mode, you would see RRUUNN on the display.

2.1.3 Line Feed

Your computer does not generate line feed control characters. It expects your terminal to do this each time a Carriage Return (Control D) is sent. If your terminal has an Auto Line Feed option, use it. Otherwise, you will have to program the interface software to insert line feeds.

2.1.4 Upper Case

Your computer expects all alphabetic characters to be in upper case. If your terminal has an Upper Case Only option, use it. Of course, the interface software can also be programmed to convert all lower case characters to upper case. An example of how to do so can be found in the driver input routine.

2.1.5 Parity

Data transfer usually is so reliable on asynchronous links that no parity generation or checking is needed, and the standard driver default command therefore selects no parity. Whichever parity option you choose (even, odd, or none), however, the terminal must be set to match the ACIA command.

2.2 RAM JUMPER

Directly beneath U9 (toward the connector fingers) are two jumper pads, labeled RAM above. If you plan to use RAMs, solder a piece of bus wire (supplied with the board) into these two pads. (If you switch from RAMs back to ROMs, you will need to remove the jumper.)

2.3 BUILDING A CABLE

Because peripheral connectors vary widely, no attempt has been made to supply the 7710 with a universal cable. You will need to construct your own cable, with a male DB-25S at one end and the appropriate connector for your peripheral at the other. Section 5.6 illustrates which RS-232-C lines should be tied to which connector pins; unsupported lines need not be connected.

2.4 CARD INSTALLATION

Now you're ready to install the interface card in the computer. First of all, align pin 1 of the I/O cable connector with pin 1 of the mating connector on the PC board. Pin 1 can be identified by the outside stripe on the cable, and by a triangular mark or other tick on the connector. When all pins are properly aligned, push down firmly on the connector until you can no longer see the metal pins. Gently fold the ribbon cable on the diagonal toward the ROMs/RAMs. Crease the fold only slightly; too much crease might fatigue and break the the wires. Now gently fold the ribbon back under itself, so that the back panel connector points to the right of the board. The slack in the cable is needed for strain relief. The card is now ready to be inserted into the computer.

```
*****
*
*  WARNING:  Do not remove the computer  *
*  cover if the power cord is plugged  *
*  in.  You could injure yourself or  *
*  damage your computer.               *
*
*****
```

Place the computer directly in front of you. Remove the top cover by placing the palms of your hands on the back edge of the computer, with your fingers hanging over the rear. Curl your fingers around the rear edge until you feel a ridge at your fingertips. Gently but firmly pry up until you hear two distinct pops. Don't lift the cover any farther. Slide it to the rear to remove it from the computer. Inside, toward the rear of the computer, you will see eight 50-pin connectors, numbered 0-7 from left to right. Place the CCS Asynchronous Serial Interface card in any of these connectors except #0, which is reserved for other use. We suggest that you use slot #2 if it is not already occupied. Insert the card by holding it and the cable so that the component side of the card is to the right and the cable assembly is to the rear. Align the card edge into the chosen connector; then gently push the card down until it is firmly seated. Slide the cable assembly through the nearest back panel slot and replace the cover on the computer. Plug one end of your signal cable (see section 2.3) to the external connector and the other end to your peripheral. Plug in the power cord, and you're ready to test the module.

2.5 CHECKOUT

Your 7710 has been fully tested, but you may for various reasons wish to test it yourself. The simple tests described in this section test most of the circuitry of the 7710. (We assume that 7710 is in slot #2. If it is not, you will have to modify the tests accordingly.)

Please note that ALL 7710 MODULES ARE SHIPPED WITH ROMS. Unless you remove these ROMs and substitute RAMs, you will want to run Test 1 and not Test 2. Test 1 is valid for substitute ROMs, but you must of course compare the screen display with a correct program listing.

2.5.1 Test 1: ROM Test

This test displays the contents of the ROMs on the CRT screen, verifying that the ROMs can be read by the computer, and allowing you to compare the contents with the program listing provided in Chapter 3.

- a. Reset your computer.
- b. Type in C200L (CR).
- c. Compare program listing to the TV display.
- d. When you run out of screen display, type in L (CR).
- e. Repeat c and d until all 256 bytes of ROM have been read.

f. If problems result, compare the hexadecimal values of the memory locations. The ROMs may be reversed on your board. If not, see your CCS dealer.

Note: Your computer's disassembler cannot recreate any assembler pseudo-operation codes, such as ORG or EQU. Occasionally, use of the ORG instruction could hide an instruction from the disassembler. For instance, the code:

```
BCS      *
ORG      *-1
SEC
```

will disassemble as

```
BCS      *+$38
```

Watch for this kind of programming trick when comparing the listings. It is valid code, but may make you think you have bad ROMs. Programming tricks such as this are used to conserve memory in tight situations.

2.5.2 Test 2: RAM Test

This test verifies that you can read from and write to all locations of the program RAMs. (Be sure you've installed the RAM jumper.) A 256-byte segment of your computer's firmware is copied into the RAMs, then the copy is compared to the

original. Errors are displayed on the screen.

- a. Reset your computer.
- b. Type in C20C<F000.F0FFM (CR).
- c. Type in C200<F000.F0FFV (CR).
- d. A * should appear almost immediately on the screen if all is OK. If it doesn't, see your CCS dealer.

2.5.3 Test 3: Serial Data Loop Test

This test checks out the ACIA, the clock, and the line drivers. It does this by sending out a known byte of data, looping it back to the receive section, reading the data, and comparing the result. For this test, you will need a "loop-back" test fixture. To make one, take a standard male DB-25S socket and wire pins 2 and 3 together, pins 4 and 5 together, and pins 8 and 20 together. This fixture allows transmitted data to be looped back into the ACIA receiver.

- a. Disconnect the signal cable.
- b. Plug the loop-back test fixture onto the end of the I/O cable.
- c. Ensure that the External baud rate is NOT selected.

- d. Turn on, Reset your computer.
- e. Type in COA0:03 (CR) to reset the ACIA.
- f. Type in COA0:11 (CR) to initialize the ACIA.
- g. Type in COA1:55 (CR) to write an alternating bit pattern.
- h. Type in COA1 (CR) to read the received data.
- i. Compare the display with what was sent.
- j. Repeat Steps g, h, and i using AA in place of 55.
- k. Repeat Steps f through i using different baud rates, ACIA commands, and data patterns until you are satisfied that the interface works properly. If you have problems, see your CCS dealer.

After completing Test 3, turn off the power, disconnect the test fixture, and reconnect the peripheral. If you changed the baud rate, set it to the correct rate. If you have ROMs on your board, you are now ready to use your CCS 7710 Asynchronous Interface. If you installed RAMs, you are now ready to develop your driver routines.

CHAPTER 3

INTERFACE SOFTWARE/FIRMWARE

The CCS Asynchronous Serial interface obtains its flexibility by balancing hardware and software. The hardware takes care of most tasks which remain the same regardless of application. The software performs the application-unique tasks. Because applications vary so greatly, it is impossible to create software which is everything to everybody. The 7710 includes a standard driver which should meet many users' needs. Examine it carefully. If it does not fit your needs, you will need to write your own driver. Sections 3.1-3.3.2 contain the information necessary for you to do so.

3.1 ACIA REGISTERS

Your computer dedicates 16 memory addresses to each peripheral connector slot (except #0) for memory-mapped input/output. These 16 addresses are above and

beyond the 256 dedicated program memory addresses. The I/O addresses are located at $\$C0xy$, where $x = 8 + n$, $n =$ the peripheral slot number (1-7), and $y =$ the specific address (1-F). In our specific case,

$\$C0x0$ (WR) = ACIA command register
 $\$C0x0$ (RD) = ACIA status register
 $\$C0x1$ (WR) = ACIA transmit register
 $\$C0x1$ (RD) = ACIA receive register

NOTE: The last address digit is not decoded beyond even or odd. This means that data can be passed on any odd address within the range, while the ACIA's command/status registers can be accessed on any even address.

For a more complete discussion of ACIA command and status formats, see a 6850 ACIA data sheet.

3.1.1 ACIA Command Register

ACIA operation is controlled by one-byte commands written to the command register. The commands are defined in the table at the top of the next page.

Please note that because the baud rate generator outputs a clock 16 times the baud rate, bits 1 and 0 must be set to 01 except when the ACIA is being reset.

ACIA COMMAND REGISTER	
7654 3210	
xxxx xx00	The clock is 1x the baud rate
xxxx xx01	The clock is 16x the baud rate
xxxx xx10	The clock is 64x the baud rate
xxxx xx11	ACIA Master Reset
xxx0 00xx	7 data + Even parity + 2 stop bits
xxx0 01xx	7 data + Odd parity + 2 stop bits
xxx0 10xx	7 data + Even parity + 1 stop bit
xxx0 11xx	7 data + Odd parity + 1 stop bit
xxx1 00xx	8 data + No parity + 2 stop bits
xxx1 01xx	8 data + No parity + 1 stop bit
xxx1 10xx	8 data + Even parity + 1 stop bit
xxx1 11xx	8 data + Odd parity + 1 stop bit
x00x xxxx	Set RS-232-C CTS
	Disable transmit interrupts
x01x xxxx	Set RS-232-C CTS
	Enable transmit interrupts
x10x xxxx	Clear RS-232-C CTS
	Disable transmit interrupts
x11x xxxx	Set RS-232-C CTS
	Transmit break on transmit data
	Disable transmit interrupts
0xxx xxxx	Disable Receive Interrupts
1xxx xxxx	Enable Receive Interrupts when:
	Receiver data register full
	Receive data overrun
	DTR signal inactive

3.1.2 ACIA Status Register

The status bits, when set, mean:

ACIA STATUS REGISTER	
Bit 0	Receive data is ready for the computer.
Bit 1	The transmit register can accept data.
Bit 2	RS-232-C DTR inactive; don't send.
Bit 3	RS-232-C RTS inactive; don't send.
Bit 4	Received data improperly framed.
Bit 5	Data received before previous byte read.
Bit 6	Parity Error in received data.
Bit 7	ACIA-generated transmit or receive interrupt.

3.2 INPUT AND OUTPUT ROUTINES

Your computer looks at two page 0 locations to find out where the current keyboard handler and console output driver programs are located. The addresses are:

\$36 - \$37: console output handler

\$38 - \$39: keyboard input handler

When you type in the BASIC command, IN#n, the firmware writes a \$00 in location \$38 and a \$Cn in location \$39. (The equivalent monitor command, n[Ctrl]K, does the same thing.) This creates an effective address of \$Cn00 for the keyboard handler initialization program. The next time keyboard input is wanted, the initialization routine gets called. It must set everything up, then pass control to the input routine to actually do the input. One of the initializer's tasks is to change location \$38 to identify the input handler's correct entry point. Then, the next time input is wanted, we can go straight to the input routine. Likewise, when OUT#n (or n[Ctrl]P) is typed in, location \$36 is set to \$00 and \$37 is set to \$Cn. On the first output, control will be passed to \$Cn00 for output initialization. Location \$36 should then be set to match the output handler's entry point for all subsequent console output.

Be aware that if any peripheral control options are allowed in the program, invoking an IN#n or OUT#n command will cause the default options to be selected for both input and output unless the

options are initialized after the input-or-output initialization decision is made. The standard program waits until after this decision is made before it initializes the character counter and operating mode. This way, an IN or OUT command has no effect on the counter and does not return the program to the default mode.

Input and output by your computer is handled on a byte-by-byte basis. As a result, data can be passed between the handlers and the computer in the accumulator (A register). Input data should be left in the accumulator when control is returned to the caller. The handler can find output data in the accumulator.

The input and output routines should be called as subroutines. Control can be returned to the caller with a simple RTS (Return from Subroutine) instruction. All register contents should be saved on entry to subroutines, then restored to their original contents just before leaving (except for parameter-passing registers). This practice saves headaches and program-debugging time later on.

Several scratchpad memory locations are available. The video-display-refresh memory locations (addresses \$400-\$7FF) use only the first 120 of every 128 locations for the display data. The remaining 24 addresses can be used for other purposes. Two sets of available locations are \$6Fx and \$77x, where $x = 8 + \text{slot number}$. For most programs, these locations should be

sufficient. But one other scratch location merits identification. Address \$7F8 is often used to hold the page address of the currently-active peripheral. The page address is \$Cn, where n is the slot number.

You now have enough information to program a simple remote console interface program. Your program should consist of three parts: initialization, input, and output. For initialization, you must:

- a. Save the registers;
- b. Reset the ACIA;
- c. Initialize the ACIA with the correct command word;
- d. Establish the proper input and/or output entry point;
- e. Allocate and/or initialize any special pointers, counters, etc;
- f. Go to Step b of the appropriate I/O routine.

For input, you must:

- a. Save the registers;
- b. Check the ACIA status and wait until the input data is ready;
- c. Read the input data;
- d. Do any special data conversion as needed--e.g., lower-to-upper case;

- e. Restore the registers;
- f. Return to the caller.

For output, you must:

- a. Save the registers;
- b. Do any special preprint control (tabs, form feeds, etc.);
- c. Wait until the ACIA can take more data;
- d. Write the data to the ACIA;
- e. Do any postprint control (line or page control, insert a line feed after a carriage return, etc.);
- f. Restore the registers;
- g. Return to the caller.

As you can see, several tasks are common to all the routines. To stretch 256 bytes of space as far as possible, you should make as much code as possible common. Since you can't predict what absolute addresses will contain this code, you can't create any subroutine calls. Thus you must use relative code throughout. Unused status flags can be used to indicate the entry point. The standard driver uses the V (overflow) flag to indicate initializing or not, and the C (carry) flag to indicate input or output. This allows you to use common segments of code, then branch to the task-unique code. After the flags have

served their purposes, they can be reused to indicate a tab in progress or other function.

The flow charts and program listing for the standard driver at the end of this chapter offer examples of the handling of specific driver tasks. We encourage you to use as much of the standard code as suits your application.

3.3 LOADING YOUR DRIVER INTO RAMS

If you selected RAMs, they must be loaded every time you turn your computer on and want to use the interface. The following procedure was devised for floppy disk systems; if you use some other storage media, you'll need to devise your own scheme.

3.3.1 Saving the Driver on Disk

The first chore is to get the interface software initially into memory. The firmware miniassembler works nicely for this; see your Red Book for details on how to use it. Assemble the driver directly into the interface's memory. For instance, if the interface is in slot #1, use address \$C100 as the base address. After you have assembled your driver into memory, save a copy of it on disk. To do this, first move a copy down into the "lower 32". Your disk

can only load and save programs from the lower 32K of memory. Location \$A00 is a good spot for the copy; it will not interfere with the Integer BASIC or the Disk Operating System (DOS). This Monitor command performs the move nicely:

```
*A00<C100.C1FFM(cr)
```

Next, transfer control over to >BASIC under the DOS. If the DOS is already in memory, just type in:

```
*3DOG
```

Otherwise, do a disk boot:

```
*6[cntrl]P(cr)
```

Finally, you are ready to actually save the driver on disk:

```
>BSAVE ASYN1.0,A$A00,L$100(cr)
```

Your driver software is now saved on your disk with a file name of ASYN1.0. You are now ready to check out the driver and modify it as necessary. Don't forget to save a copy after each modification. There's nothing more frustrating than to try to check out a routine only to have it bomb out and erase itself in the process. When you're happy with the routine, save it one last time.

3.3.2 Power-On Loading of the Driver

Two methods of loading the software from disk are outlined here: a) using direct commands, and b) under program control.

a. Direct Commands:

To load the driver with direct commands, perform the following sequence:

1. Boot in the DOS:

*6(ctrl P)(cr)

2. Read in the driver file:

>BLOAD ASYN1.0(cr)

3. Return control to the monitor:

>CALL-155(cr)

4. Finally, upload the driver into the interface's RAM. Assuming that the interface is in slot #2:

*C200<A00.AFFM(cr)

b. Loading the program under program control:

This alternate method combines steps 2 and 4 above into one automated step. A simple >BASIC program to perform this is:

```
10 INPUT "ASYNC INTERFACE SLOT IS: ",S
20 IF S<1 OR S>7 THEN GOTO 10
30 DEST = -16384 + 256 * S
40 PRINT "(ctrl D)BLOAD ASYN1.0,A$A00"
50 FOR I = 0 TO 255
60 POKE DEST + I, PEEK (2560 + I)
70 NEXT I
80 END
```

Assuming that this program has been saved on disk under the file name of ASYN, all you have to do now is:

1. Boot in the DOS:

*6(ctrl P)(cr)

2. Execute the ASYN program:

>RUN ASYN(cr)

3. Answer the question that appears:

ASYNC INTERFACE SLOT IS: ?2(cr)

3.4 CALLING INPUT

The programs you install in the ROMs/RAMs won't do any good unless control can be passed to them for input or output. To do this, type in (n = slot number):

```
IN#n (> or] BASIC)
n(ctrl K) (Monitor)
```

Either of these commands will cause your computer to go to the installed input program on the card for all subsequent input to the computer. On the first input, the ACIA will be initialized if the interface program works like the standard program.

3.5 CALLING OUTPUT

To cause all output to be controlled by the programs on the card, type in one of the following commands (n = slot number):

```
PR#n    (>or ] BASIC)
n(ctrl P) (Monitor)
```

All subsequent output from the computer will now be routed to the card's interface program.

3.6 CHANGING DEFAULT PARAMETERS

There are two default parameters in the standard program which can be changed AFTER an IN#n or OUT#n command has been executed.

3.6.1 ACIA Operating Mode Command

The default value for the ACIA operating mode command is \$11, which specifies an 8 bit word, 2 stop bits, no parity, and no interrupts. To change the operating mode, POKE the desired command into location $16256 + 16*n$ ($\$C080 + \$n0$), where n is the slot number. Remember, though, that any subsequent IN# n or OUT# n command will re-command the ACIA to the default value.

3.6.2 Characters Per Line

The sample driver will automatically initiate a carriage return, line feed sequence after every 255 characters have been printed. If you want some other number of characters per line, simply POKE your value into location $\$5F8$ ($1528d$) + the slot number. Make sure it is in the range $0 < CPL \leq 255$.

3.7 HAVING TROUBLE?

If your computer and peripheral are not talking to each other, you may be encountering one of the following common problems:

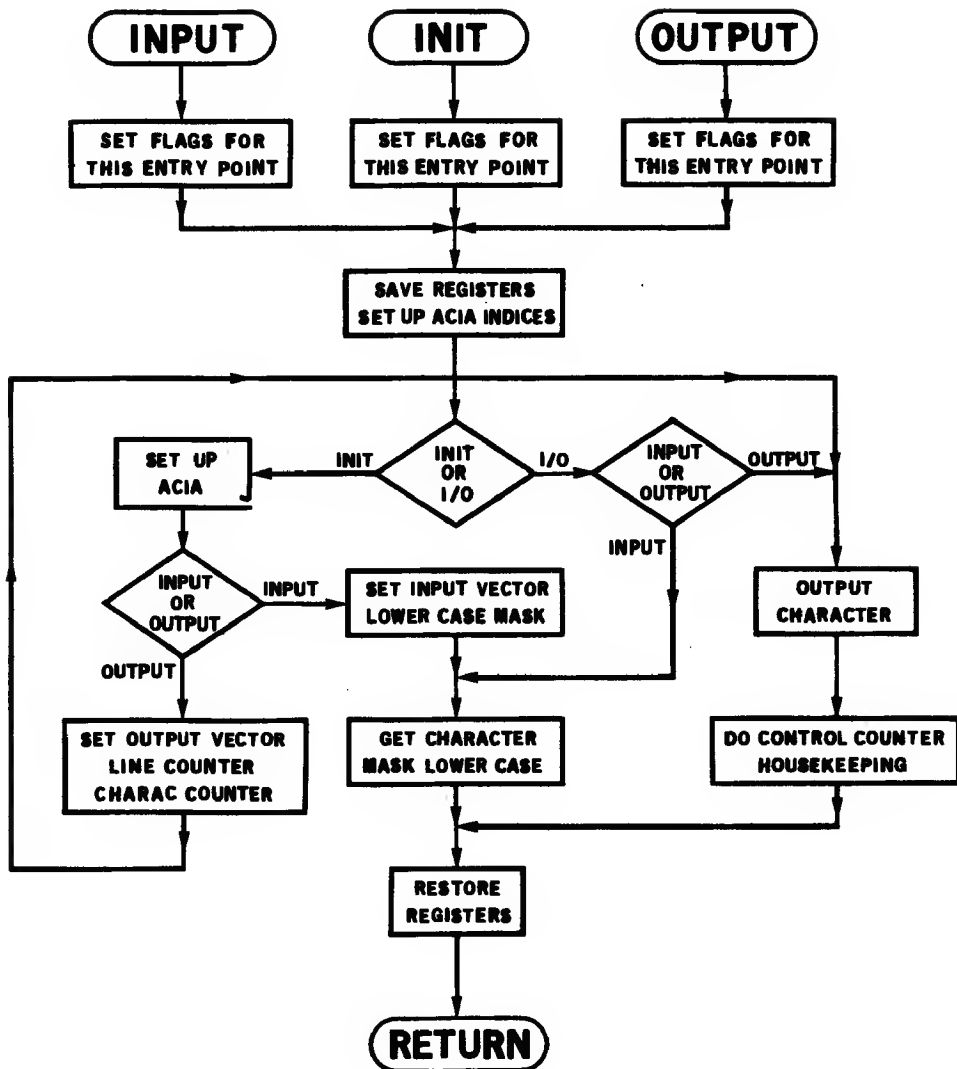
PROBLEM Your computer is not sending data.

SOLUTION For the 7710 to send data, the RTS line, pin 4, must be high. Install a handshake line from the receiving device or a jumper from pin 4 to pin 6 or 8 of the RS-232-C connector. If you use a jumper, data will be sent at the proper baud rate but could overrun the receiving buffer, causing data loss. If this happens, use a slower baud rate.

PROBLEM Your computer is not receiving data.

SOLUTION For the 7710 to receive data, the DTR line must be high. Install a handshake line from the sending device or a jumper from pin 20 to pin 6 or 8 of the connector.

3.8 FLOW CHART



Address	Code	Comments
0000	LOCASE	EQU
0001	CHCNT	EQU
0002	CMD	EQU
0003	STATUS	EQU
0004	DATA	EQU
0005	WAIT	EQU
0006	RETURN	EQU
0007	;	The common code
0008	;	
0009	INIT:	BIT
0010	OUTEP:	BVS
0011	INEP:	CLC
0012	COM:	DFB
0013		SEC
0014		CLV
0015		PHA
0016		TXA
0017		PHA
0018		TYA
0019		PHA
0020		PHP
0021		SEI
0022		JSR
0023		TSX
0024		LDY
0025		PLA
0026		PLA
0027		PLA
0028		TXS
0029		PHA
0030	2C	
0031	70	
0032	04	
0033	18	
0034	B0	
0035	38	
0036	B8	
0037	48	
0038	8A	
0039	48	
0040	98	
0041	48	
0042	08	
0043	78	
0044	20	
0045	BA	
0046	BC	
0047	68	
0048	68	
0049	68	
0050	9A	
0051	01B	
0052	01C	
0053	CB	
0054	FF	
0055	00	
0056	01	
0057		
0058		
0059		
0060		
0061		
0062		
0063		
0064		
0065		
0066		
0067		
0068		
0069		
0070		
0071		
0072		
0073		
0074		
0075		
0076		
0077		
0078		
0079		
0080		
0081		
0082		
0083		
0084		
0085		
0086		
0087		
0088		
0089		
0090		
0091		
0092		
0093		
0094		
0095		
0096		
0097		
0098		
0099		
0100		
0101		
0102		
0103		
0104		
0105		
0106		
0107		
0108		
0109		
0110		
0111		
0112		
0113		
0114		
0115		
0116		
0117		
0118		
0119		
0120		
0121		
0122		
0123		
0124		
0125		
0126		
0127		
0128		
0129		
0130		
0131		
0132		
0133		
0134		
0135		
0136		
0137		
0138		
0139		
0140		
0141		
0142		
0143		


```

0043 38      SEC
0044 90 31      BCC
0046          ;Fall through next branch
0046          ;Go to normal output
0046
0046      ; Input Routine
0046
0046      ;This routine expects no parameters from calling routines.
0046      ;It waits until data has been typed in from the keyboard, and
0046      ;then returns that data in the accumulator to the caller. It
0046      ;ignores all line feeds and converts all lower case letters
0046      ;to upper case (if the locase mask = #$20).
0046
0046      INPUT:      LDA
0049 4A      LSR
004A 90 FA      BCC
004C 68      PLA
004D B9 81 C0   LDA
0050 09 80      ORA
0052 48      PHA
0053 C9 8A      CMP
0055 F0 EF      BEQ
0057 68      PLA
0058 C9 E0      CMP
005A 90 74      BCC
005C 5D 38      EOR
005F B0 6F      BCS
0061          ;Output Initialization
0061          ;
0061
0061      OINIT:      LDA
0063 85 36      STA
0065 A9 FF      LDA
0067 9D 38 05     STA

```

;Fall through next branch
 ;Go to normal output

 ; Input Routine

 ;This routine expects no parameters from calling routines.
 ;It waits until data has been typed in from the keyboard, and
 ;then returns that data in the accumulator to the caller. It
 ;ignores all line feeds and converts all lower case letters
 ;to upper case (if the locase mask = #\$20).

 INPUT: ;Get ACIA status
 ;Isolate Rx Rdy bit
 ;Loop until data is ready
 ;Get rid of data on stack top
 ;Read the new data
 ;Set Bit 7 for normal video
 ;Save data on stack
 ;Ignore Line Feeds

 STATUS,Y ;See if it is lower case
 A ;Go finish up if not
 INPUT ;Convert to upper if mask = \$20
 DATA,Y ;Now go finish up
 #\$80
 \$LNFD
 INPUT

 #\$E0
 DONA
 LOCASE,X
 DONA

 ;Output Initialization
 ;
 ;
 OINIT: ;Set normal output entry point vector
 #5
 CSWL
 #MAXCHR
 CPL,X
 ;Set defaults

Address	Code	Comments
006A	A9 36	
006C	9D B8 06	LDA #MAXLN
006E		STA LPP,X
006F		;; Output Routine
006F		;;
006F		;; This routine does the actual output of the data. It expects
006F		;; to find the data for output on the top of the stack (where
006F		;; the common code put it).
006F		;;
006F		OUT:
006F	BD B8 06	LDA CHCNT,X ;See if tab wanted
0072	C5 24	CMP CH
0074	B0 03	BCS OUTA ;Branch if no tab
0076	A9 A0	LDA #SPACE ;Space out to column
0078	48	PHA ;Save on stack
0079	68	PLA ;Retrieve character
007A	48	PHA ;Resave it
007B	08	PHP ;Save Tab Status
007C	C9 95	CMP #FF ;Check for form feed
007E	F0 04	BEQ CHCTI ;Branch if not
0080	29 60	AND #\$60 ;Test for other cntrl char
0082	F0 03	BEQ OUTB ;Skip counter Increment
0084	FE B8 06	INC CHCNT,X ;Bump counter
008D	28	PLP ;Reget tab flag
009E	B9 80 C0	LDA STATUS,Y ;Get ACIA status
008B	29 03	AND #3 ;Isolate Tx buffer bit
008D	F0 F9	BEQ OUTC ;Wait if not ready
008F	29 01	AND #\$1 ;Check for input char
0091	D0 46	BNE INCHR ;Get char if there is one
0093	68	PLA ;Get data for output
0094	99 81 C0	STA DATA,Y ;Output it
0097	90 D6	BCC OUT ;Branch if tab
0099	70 01	BVS LF ;Branch if self generating character

```

009B 48      PHA
009C C9      CMP
009E F0      BEQ
00A0 E9      SBC
00A2 D0      BNE
00A4 DE      DEC
00A7 30      BMI
00A9
00A9
00A9
00A9
00A9
00A9
00A9 68      PLA
00AA BA      TSX
00AB E8      INX
00AC E8      INX
00AD E8      INX
00AE 9D      STA
00B1 68      PLA
00B2 A8      TAY
00B3 68      PLA
00B4 AA      TAX
00B5 68      PLA
00B6 60      RTS
00B7
00D7
00B7
00B7 E9 06
00B9 D0 12
00BB 9D B8 06
00BE 85 24

LF:
BS:
CR:
CRA:

#LNFD
DONE
#BKSP
OR
CHCNT,X
CRA

; Resave character
; See if line feed

; See if back space
; No, branch
; Yes, adjust counter
; Disallow negative count

; Final Common Code
; This part of the code restores the registers and returns to
; the caller. It is used by all of these routines.
; DONE:
; DONA:
; Set the stack straight
; Modify A register value in
; Stack to insure it is
; Restored to the right value

; Restore registers

; Done!

; Rest of output code
; CR:
; CRA:
; See if a Carriage Return
; No, branch
; Zero out counter
; and tab pointer

```

```

00C0          #C0          ;Wait for print head to return
00C2          WAIT
00C5          #LNFD
00C7          RETURN
00CA          ;Make a line feed
00CB          ;V=1 for self generating character
00CD          ;C=1 for no tab
00D0          ;Always branch
00D3          ;End Of line yet?
00D5          OUTD
00D7          CHCNT,X
00D9          CPL,X
00D9          DONE
00D9          #CARRET
00D9          SELF
00D9          ;No, done
00D9          ;Yes, get a Carriage Return
00D9          ;Process it

;          Cntrl S Interrupt Code
;
;This routine processes the cntrl S (DC3) to interrupt
;sending. Any character will cause sending to resume.
;
;INCHR
00D9          PHP
00DA          LDA          DATA,Y
00DD          AND          #$7F
00DF          CMP          #$13
00E1          BEQ          WAIT1
00E3          PLP
00E4          BNE
00E6          LDA          OUTC
00E9          AND          STATUS,Y
00EB          BEQ          $1
00ED          LDA          WAIT1
00F0          PLP          DATA,Y
00F1          BNE          OUTC
00F1          C0          81
00F1          C0          7F
00F1          C0          13
00F1          C0          03
00F1          C0          28
00F1          C0          A2
00F1          C0          D0
00F1          C0          B9
00F1          C0          80
00F1          C0          01
00F1          C0          29
00F1          C0          F0
00F1          C0          09
00F1          C0          B9
00F1          C0          81
00F1          C0          28
00F1          C0          95

```

CHAPTER 4

TECHNICAL INFORMATION

4.1 SPECIFICATIONS

SIZE: 5" l x 2.75" h x .75" w (max)

WEIGHT: less than 5 oz. (w/ cable)

SYSTEM INTERFACE:

Internal: APPLE II
Peripheral slots 1 through 7

External: EIA RS-232-C (1969)
Configuration A thru E
Primary Circuits Only
Interface Type: DCE
Failsafe Input Circuits
(Shorts and Opens)

DATA TRANSFER MODE: Asynchronous Serial
7 or 8 Data Bits
Odd, Even, or No Parity Bits
1 or 2 Stop Bits

DATA TRANSFER RATES:	50 baud	75 baud	110 baud
(Clock = 16 x baud)	134.5 baud	150 baud	200 baud
	300 baud	600 baud	1200 baud
	1800 baud	2400 baud	4800 baud
	9600 baud	19200 baud	External

PROGRAM MEMORY: 256 Bytes ROM; may be
replaced by RAM (Static: 2112's)
ROM/PROM Auto Powered Down

REQUIRED POWER: +5vdc
+12vdc
-12vdc

FEATURES: Supports Daisy Chain Interrupts
with On-Board Arbitration Logic
Allows DMA Daisy Chain Pass-Through
Crystal-Controlled Baud Rates
Baud Rates Switch Selectable
Glass Epoxy (FR-4) PC Board
Gold Plated Connector Fingers
Solder Mask Both Sides of Board
Component Silkscreen

ASI-1
MODEL 7710A
ASYNCHRONOUS SERIAL INTERFACE

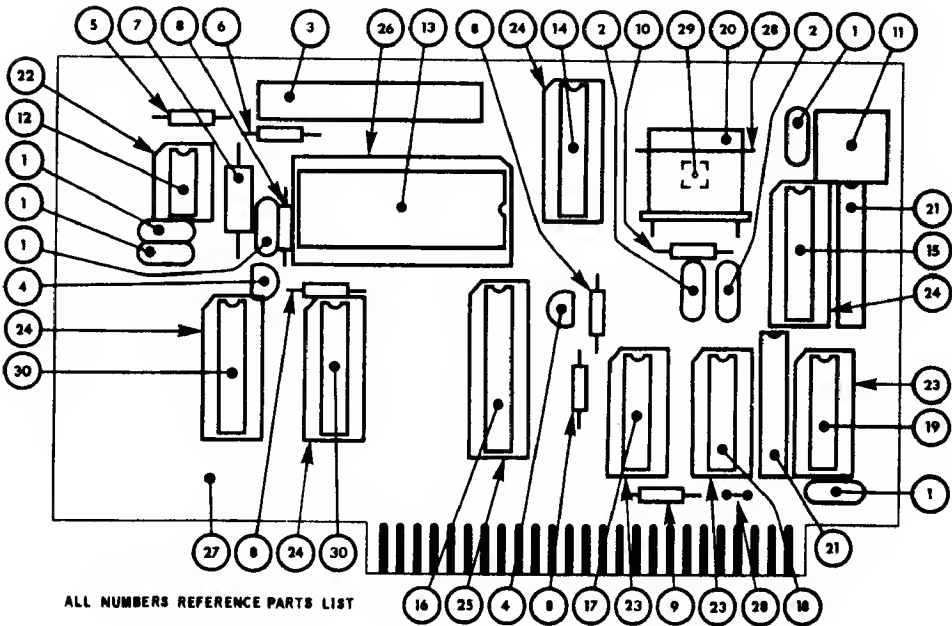
4.3 PARTS LIST

CCS 7710A (Assy 00000-7710A)

#	QTY	REF	CCS P/N	DESCRIPTION
1	5	C1-4,7	42034-21046	CAPACITOR, MONOLYTHIC .1uf, 50vdc
2	2	C5,6	42215-55605	CAPACITOR, MICA 56pf, 500vdc, 10%
3	1	J2	56004-02013	HEADER, DUAL 13 PIN
4	2	Q1,2	36100-02907	TRANSISTOR, SI; PNP GENERAL PURPOSE, PN2907
5	1	R1	40002-01005	RESISTOR, FIXED, COMP 10 ohm, 1/4W, 10%
6	1	R2	40002-06815	RESISTOR, FIXED, COMP 680 ohm, 1/4W, 10%
7	1	R3	40003-01015	RESISTOR, FIXED, COMP 100 ohm, 1/2W, 10%
8	4	R4-7	40002-02215	RESISTOR, FIXED, COMP 220 ohm, 1/4W, 10%
9	1	R8	40002-02725	RESISTOR, FIXED, COMP 2.7K, 1/4W, 10%
10	1	R9	40002-01055	RESISTOR, FIXED, COMP 1M, 1/4W, 10%
11	1	S1	27111-41010	SWITCH, DIP, 4PST
12	1	U1	30300-00150	IC, INTERFACE; 75150 DUAL RS-232-C DRIVER
13	1	U2	31100-06850	IC, DIGITAL, MOS; 6850 ACIA
14	1	U3	30300-00154	IC, INTERFACE; 75154 QUAD RS-232 RCVR
15	1	U4	31000-04702	IC, DIGITAL, CMOS; 4702 BAUD RATE GENERATOR
16	1	U7	30900-08304	IC, DIGITAL, TTL; 8304B OCTAL BUS DRVR/RCVR
17	1	U8	30000-00009	IC, DIGITAL, TTL; 74LS09 QUAD 2 IN AND O.C.
18	1	U9	30000-00136	IC, DIGITAL, TTL; 74LS136 QUAD 2 IN EX-OR O.C.
19	1	U10	30000-00003	IC, DIGITAL, TTL; 74LS03 QUAD 2 IN NAND O.C.
20	1	Y1	48132-45762	XTAL, QUARTZ 2.4576MHz, HC-6
21	2	Z1,2	40930-72726	RESISTOR NETWORK, SIP 2.7K x 7
22	1	XU1	58102-00080	SOCKET, IC; LOW PROFILE 8 PIN DIP
23	3	XU8-10	58102-00140	SOCKET, IC; LOW PROFILE 14 PIN DIP

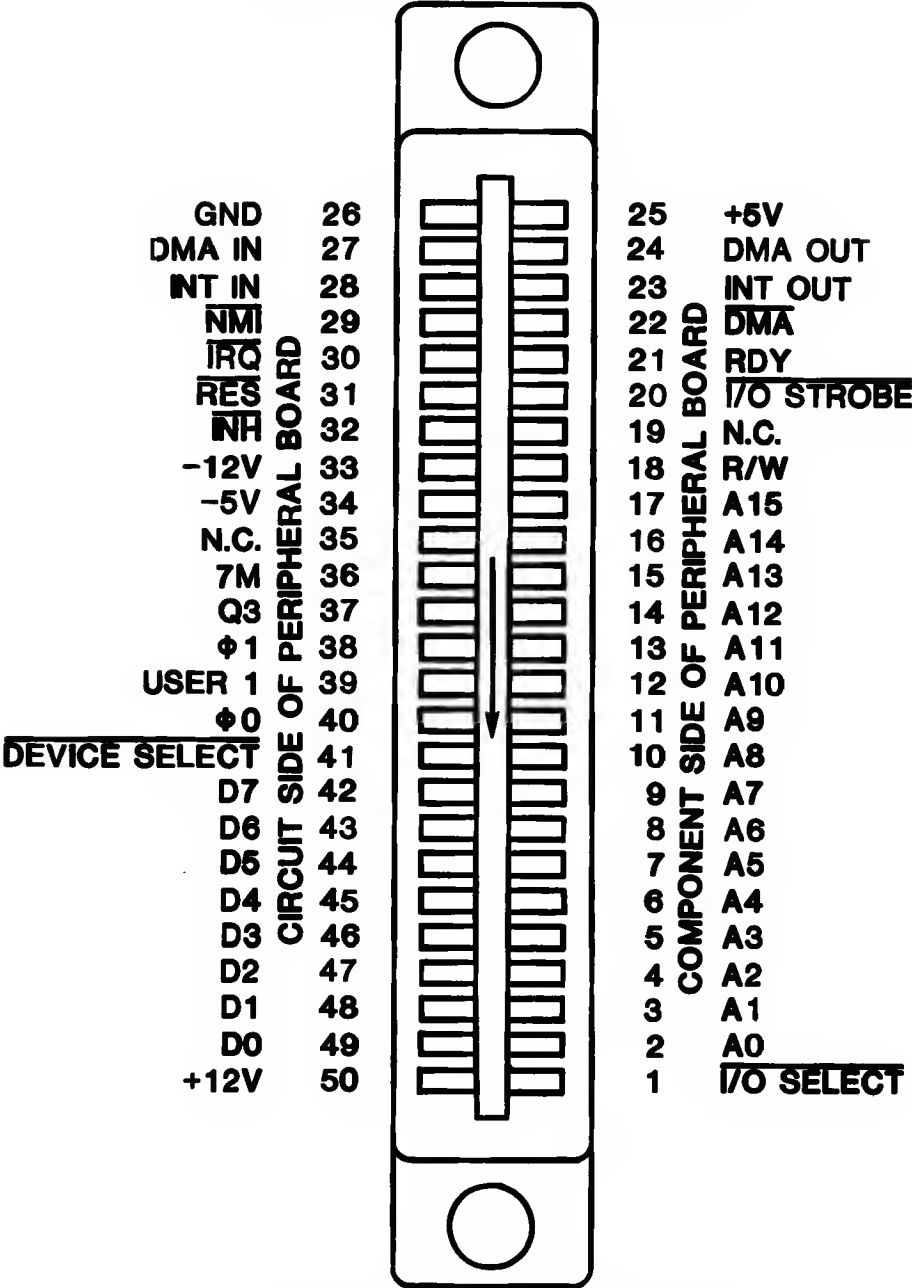
#	QTY	REF	CCS	P/N	DESCRIPTION
24	4	XU3-6	58102-00160		SOCKET, IC; LOW PROFILE 16 PIN DIP
25	1	XU7	58102-00200		SOCKET, IC; LOW PROFILE 20 PIN DIP
26	1	XU2	58102-00240		SOCKET, IC; LOW PROFILE 24 PIN DIP
27	1	-	07710-00002		BOARD, PC ASI-1, REV A
28	-	(Y1)	51000-01220		WIRE, BARE; TINNED #22 AWG
29	-	(Y1)	60003-00001		TAPE, FOAM; 2 SIDED
30	1	U5,U6	00000-7610C		IC, DIGITAL, TTL, ROM-PAK

4.4 PARTS BREAKDOWN



4.5 APPLE II I/O CONNECTOR

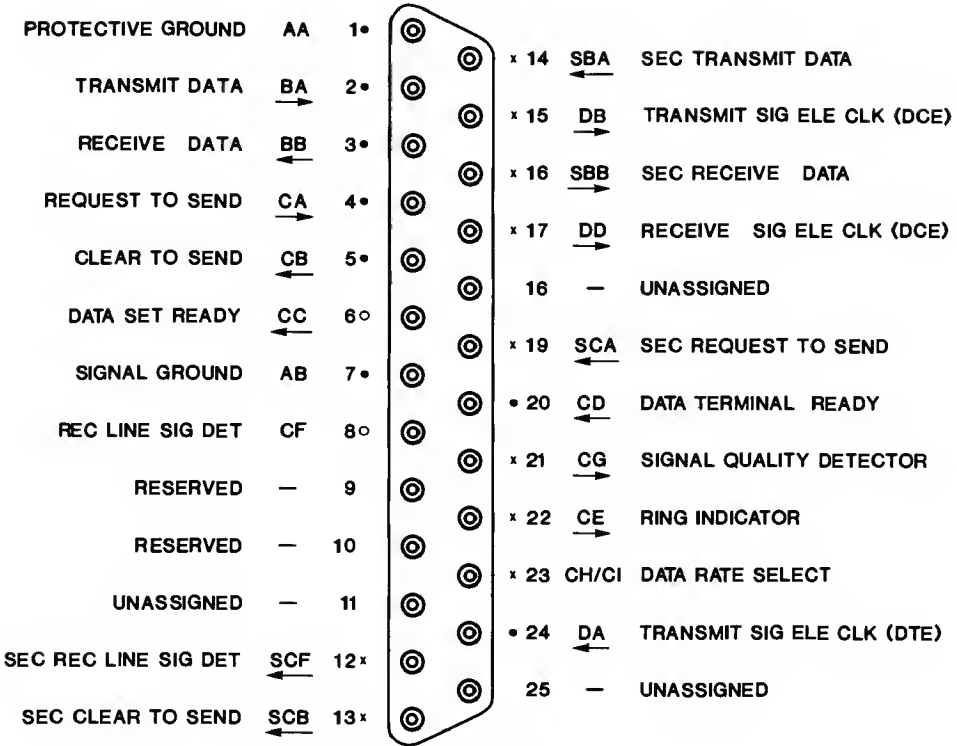
TOP VIEW
BACK OF APPLE MAIN BOARD



FRONT OF APPLE MAIN BOARD

4.6 RS-232-C DCE CONNECTOR

FRONT VIEW



DB-25S (FEMALE)

7710

- FULL ACTIVE SUPPORT
- ° PASSIVE SUPPORT
- * NOT SUPPORTED

EIA RS-232C
DCE TYPE CONNECTOR PIN ASSIGNMENT

4.7 DEFINITION OF RS-232-C INTERFACE CONFIGURATIONS

A	Transmit Only
B	Transmit Only*
C	Receive Only
D	Half Duplex; or Duplex*
E	Full Duplex
F	Primary Channel Transmit Only* / Secondary Channel Receive Only*
G	Primary Channel Receive Only / Secondary Channel Transmit Only*
H	Primary Channel Transmit Only / Secondary Channel Receive Only
I	Primary Channel Receive Only / Secondary Channel Transmit Only
J	Primary Channel Transmit Only* / Half Duplex Secondary Channel
K	Primary Channel Receive Only / Half Duplex Secondary Channel
L	Half Duplex Primary Channel / Half Duplex Secondary Channel*
L	Duplex Primary Channel* / Duplex Secondary Channel*
M	Duplex Primary Channel / Duplex Secondary Channel
Z	Special (Circuits specified by supplier)

*Note the inclusion of Request to Send in a Transmit Function, where it would not ordinarily be expected, but could indicate a non-transmit mode to the data communications equipment (DCE) to permit it to remove a line signal or to send synchronizing or framing signals as required.

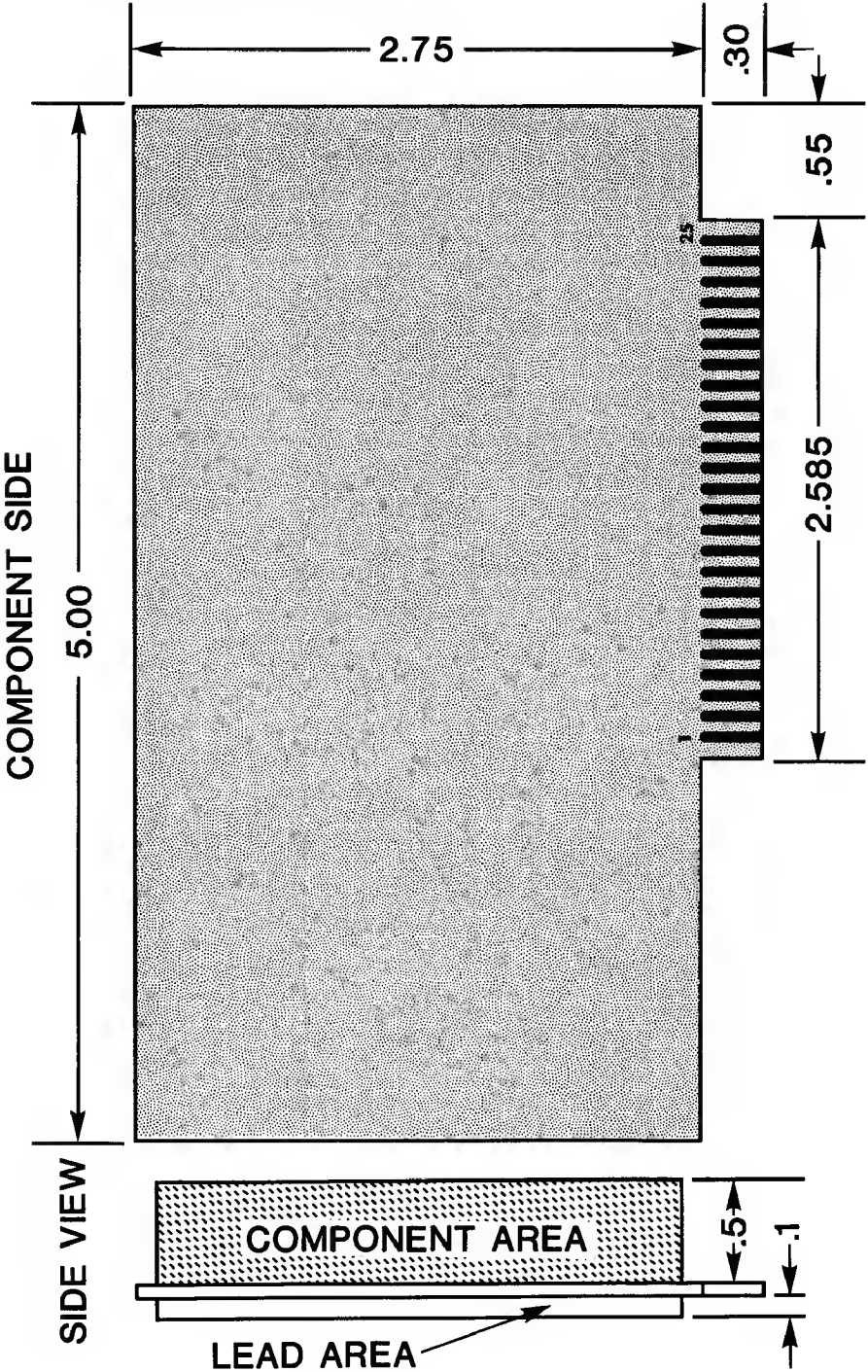
4.8 STANDARD INTERFACE CONFIGURATIONS FOR RS-232-C

Interchange Circuit

Interchange Circuit														
	Interface Configuration													
	A	B	C	D	E	F	G	H	I	J	K	L	M	Z
AA	-	X	X	X										-
AB	X	X	X											X
BA														O
BB														O
CA														O
CB														O
CC														O
CD														O
CE														O
CF														O
CG														O
CH/CI														O
DA/DB														O
DD	T	T		+										O
SBA														O
SBB														O
SCA														O
SCB														O
SCF														O

Notes: Upper case indicates a line supported by the CCS 7710.
Lower case indicates a line not supported by the CCS 7710.
X = Basic interchange circuits, all systems
T = Additional interchange circuits required for synchronous channel
S = Additional interchange circuits required for switched service
O = Specified by supplier as required
- = Optional; supported by the CCS 7710

4.9 BOARD DIMENSIONS



APPENDIX A

LIMITED WARRANTY

California Computer Systems (CCS) warrants to the original purchaser of its products that its CCS assembled and tested products will be free from materials defects for a period of one (1) year, and be free from defects of workmanship for a period of ninety (90) days.

The responsibility of CCS hereunder, and the sole and exclusive remedy of the original purchaser for a breach of any warranty hereunder, is limited to the correction or replacement by CCS at CCS's option, at CCS's service facility, of any product or part which has been returned to CCS and in which there is a defect covered by this warranty. CCS will correct any defect in materials and workmanship free of charge if the product is returned to CCS within ninety (90) days of original purchase from CCS; and CCS will correct defects in materials in its products and restore the product to an operational status for a labor charge of \$25.00, provided that the product is returned to CCS within one (1) year. All such returned products shall be shipped prepaid and insured by original purchaser to:

Warranty Service Department
California Computer Systems
250 Caribbean Drive
Sunnyvale, California
94086

CCS shall have the right of final determination as to the existence and cause of a defect, and CCS shall have the sole right to decide whether the product should be repaired or replaced.

This warranty shall not apply to any product or any part thereof which has been subject to

(1) accident, neglect, negligence, abuse or misuse;

(2) any maintenance, overhaul, installation, storage, operation, or use, which is improper; or

(3) any alteration, modification, or repair by anyone other than CCS or its authorized representative.

THIS WARRANTY IS EXPRESSLY IN LIEU OF ALL OTHER WARRANTIES EXPRESSED OR IMPLIED OR STATUTORY INCLUDING THE WARRANTIES OF DESIGN, MERCHANTABILITY, OR FITNESS OR SUITABILITY FOR USE OR INTENDED PURPOSE AND OF ALL OTHER OBLIGATIONS OR LIABILITIES OF CCS. To any extent that this warranty cannot exclude or disclaim implied warranties, such warranties are limited to the duration of this express warranty or to any shorter time permitted by law.

CCS expressly disclaims any and all liability arising from the use and/or operation of its products sold in any and all applications not specifically recommended, tested, or certified by CCS, in writing. With respect to applications not specifically recommended, tested, or certified by CCS, the original purchaser acknowledges that he has examined the products to which this warranty attaches, and their specifications and descriptions, and is familiar with the operational characteristics thereof. The original purchaser has not relied upon the judgement or any representations of CCS as to the suitability of any CCS product and acknowledges that CCS has no knowledge of the intended use of its products. CCS EXPRESSLY DISCLAIMS ANY LIABILITY ARISING FROM THE USE AND/OR OPERATION OF ITS PRODUCTS, AND SHALL NOT BE LIABLE FOR ANY CONSEQUENTIAL OR INCIDENTAL OR COLLATERAL DAMAGES OR INJURY TO PERSONS OR PROPERTY.

CCS's obligations under this warranty are conditioned on the original purchaser's maintenance of explicit records which will accurately reflect operating conditions and maintenance performed on CCS's products and

establish the nature of any unsatisfactory condition of CCS's products. CCS, at its request, shall be given access to such records for substantiating warranty claims. No action may be brought for breach of any express or implied warranty after one (1) year from the expiration of this express warranty's applicable warranty period. CCS assumes no liability for any events which may arise from the use of technical information on the application of its products supplied by CCS. CCS makes no warranty whatsoever in respect to accessories or parts not supplied by CCS, or to the extent that any defect is attributable to any part not supplied by CCS.

CCS neither assumes nor authorizes any person other than a duly authorized officer or representative to assume for CCS any other liability or extension or alteration of this warranty in connection with the sale or any shipment of CCS's products. Any such assumption of liability or modification of warranty must be in writing and signed by such duly authorized officer or representative to be enforceable. These warranties apply to the original purchaser only, and do not run to successors, assigns, or subsequent purchasers or owners; AS TO ALL PERSONS OR ENTITIES OTHER THAN THE ORIGINAL PURCHASER, CCS MAKES NO WARRANTIES WHATSOEVER, EXPRESS OR IMPLIED OR STATUTORY. The term "original purchaser" as used in this warranty shall be deemed to mean only that person to whom its product is originally sold by CCS.

Unless otherwise agreed, in writing, and except as may be necessary to comply with this warranty, CCS reserves the right to make changes in its products without any obligation to incorporate such changes in any product manufactured theretofore.

This warranty is limited to the terms stated herein. CCS disclaims all liability for incidental or consequential damages. Some states do not allow limitations on how long an implied warranty lasts and some do not allow the exclusion or limitation of incidental or consequential damages so the above limitations and exclusions may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

